REMARKS/ARGUMENTS

In response to the objection to the specification, the specification has been amended on page 9, line 10.

In response to the objection to the claims, claims 7 and 13 have been amended.

On page 3 of the Official Action, claims 1, 4-7, 9, and 15-18 were rejected under 35 U.S.C. 103(a) as being unpatentable over "PAE36 and Linux Virtual Memory System" in view of Emmes (U.S. Patent 6,981,125). Applicant respectfully traverses and submits that the invention of claims 1, 4-7, 9, and 15-18 would not have been obvious from PAE36 in view of Emmes, because Emmes would not have motivated one of ordinary skill to modify PAE36 in the fashion required to reconstruct the applicant's invention.

The Official Action cites PAE36, page 3, 2nd paragraph: This says:

Logical Memory

Despite the fact that a computer may now physically access up to 64GB (2^{36} bytes) of memory, each process still receives at maximum 3GB of *virtual* address space for use. This limit is imposed by the use of 32-bit pointers, and the fact that the kernel reserved the upper 1GB of the 32-bit address space for itself [USE]. Nevertheless, it is possible to work around this limit, for instance, data may be distributed over several processes. In effect, each process could manage a 3GB "chunk" of memory, and data could be passed between processes using standard IPC or shared memory.

Page 4 of the Official Action refers to "shared memory which is equivalent to common memory." However, such an equivalence is not taught in PAE36. For example, a conventional method of passing data between processes is the mechanism employed in subroutine calls, namely, by pushing processor context onto the processor stack. In a similar fashion, a conventional method of passing parameters between program routines and returning results is by the calling routine pushing processor context and parameters onto the stack, and the called routine popping the parameters from the stack. For example, this is done transparently to a programmer of a high-level programming language, but the details of passing parameters and returning a result should be well understood by anyone attempting to write a machine language subroutine. Thus, one of ordinary skill could understand "data could be passed between processes using standard IPC or shared memory" as disclosing nothing more than communication between processes using processor registers and the processor stack.

Page 4 of the Official Action recognizes that: "PAE36 fails to teach at least one of the virtual memory spaces includes a chunk of physical memory that is not included in any other of the plurality of virtual memory spaces." Therefore, page 4 of the Official Action says: "Emmes teaches at least one of the virtual memory spaces includes a chunk of physical memory that is not included in any other of the plurality of virtual memory spaces (FIG. 7A and 8; column 1, lines 9-12; column 9, lines 3-16; column 10, lines 6-18). It is true that FIG. 7A of Emmes shows a pair of virtual addresses spaces including a shared or common portion of real storage and private space, but the applicants' claim 1 is more specifically directed to "the chunk of physical memory

that is not included in any other of the plurality of virtual memory spaces being assigned for use by a software module, and the digital computer being programmed for using the common physical memory for communication of parameters to and results from the software module."

For combining PAE36 with Emmes, page 4 of the Official Action says: "Emmes states that "the present invention provides a way to support sharing without allocating a new band of virtual storage across all address spaces for management purposes" (column 20, lines 54-57.) However, it is not seen how this would have motivated one of ordinary skill in the art to combine and modify the fair teachings of PAE36 and Emmes to arrive at the invention of applicants' claim 1.

For example, the applicant teaches that multiple levels of indirection in the address translation of a physical address extension (PAE) feature of a processor may cause a loss of performance unless there is an appropriate assignment of virtual memory spaces to well-defined or well-contained software modules executed by the processor. (See applicant's specification, page 3, lines 14-17.) In contrast, the Emmes column 20, lines 54-57 does not appear to relate to assignment of virtual memory spaces to well-defined or well-contained software modules executed by the processor, and instead appears to be directed to more efficient virtual-to-physical address translation. Applicant's claims 9 and 15, for example, further define more than one software module.

Hindsight reconstruction, using the applicant's specification itself as a guide, is improper because it fails to consider the subject matter of the invention "as a whole" and fails to consider the invention as of the date at which the invention was made. "[T]here must be some motivation,

suggestion, or teaching of the desirability of making the specific combination that was made by the applicant." In re Lee, 277 F.3d 1338, 1343, 61 U.S.P.Q.2d 1430, 1435 (Fed. Cir. 2002) (quoting In re Dance, 160 F.3d 1339, 1343, 48 U.S.P.Q.2d 1635, 1637 (Fed. Cir. 1998)). "[T]eachings of references can be combined only if there is some suggestion or incentive to do so." In re Fine, 837 F.2d 1071, 1075, 5 U.S.P.Q.2d 1596, 1600 (Fed. Cir. 1988) (Emphasis in original) (quoting ACS Hosp. Sys., Inc. v. Montefiore Hosp., 732 F.2d 1572, 1577, 221 U.S.P.Q. 929, 933 (Fed. Cir. 1984)). "[P]articular findings must be made as to the reason the skilled artisan, with no knowledge of the claimed invention, would have selected these components for combination in the manner claimed." In re Kotzab, 217 F.3d 1365, 1371, 55 U.S.P.Q.2d 1313, 1317 (Fed. Cir. 2000). See, for example, Fromson v. Advance Offset Plate, Inc., 755 F.2d 1549, 1556, 225 U.S.P.Q. 26, 31 (Fed. Cir. 1985) (nothing of record plainly indicated that it would have been obvious to combine previously separate lithography steps into one process); In re Gordon et al., 733 F.2d 900, 902, 221 U.S.P.Q. 1125, 1127 (Fed. Cir. 1984) (mere fact that prior art could be modified by turning apparatus upside down does not make modification obvious unless prior art suggests desirability of modification); Ex Parte Kaiser, 194 U.S.P.Q. 47, 48 (PTO Bd. of Appeals 1975) (Examiner's failure to indicate anywhere in the record his reason for finding alteration of reference to be obvious militates against rejection).

Regarding claim 4, page 4 of the Official Action cites Emmes col. 2, lines 1-7, which say:

In the case of the common area, all common segments are mapped by a set of common page tables which are pointed to by the appropriate entries in the

> segment tables of each address space. In this common area are system (or kernel) code and control structures, together with authorized code and control structures. Page tables (and segment tables) in these prior systems are virtualized and hence consume some 8 MB of pre-allocated virtual space within each private space. In order to provide access to the common space, the segment table entries for the common segments are identical in content for each address space.

Is not understood how this passage discloses that at least one other of the virtual memory spaces is directly mapped to a bottom region of the physical memory address space and is allocated to page tables.

Regarding claim 5, page 5 of the Official Action cites Emmes column 5, lines 4-27. This passage relates to swap-out processing during which "all DAT tables which are owned by the address space and which map (indirectly) shared ranges are discarded after deregistering interest in the shared tables owned by the operating system." This passage appears to relate to switching of the virtual address translation. It is not understood how this passage discloses the applicants' claimed operations of "copying at least one parameter from a context of an application to the common physical memory, …, executing the software module for processing said at least one parameter to produce a result placed in the common physical memory, …, and copying the result from the common physical memory to the context of the application."

Regarding claims 6-7 and 16-17, page 5 of the Official Action again cites Emmes column 5, lines 4-27. It is not understood how this passage discloses that the virtual memory space of the application is directly mapped to a bottom region of the physical memory address space, nor

does it appear to disclose turning paging on and off and disabling or enabling a thread scheduler. Where the prior art references fail to teach a claim limitation, there must be "concrete evidence" in the record to support an obviousness rejection. "Basic knowledge" or "common sense" is insufficient. *In re Zurko*, 258 F.3d 1379, 1385-86, 59 U.S.P.Q.2d 1693, 1697 (Fed. Cir. 2001).

On page 6 of the Official Action, claims 2, 3, 10, and 13 were rejected under 35 U.S.C. 103(a) as being unpatentable over the PAE36 reference and Emmes and further in view of Atherton et al. (U.S. Patent 6,981,125). Applicants respectfully traverse. Claims 2 and 3 are dependent claims depending from claim 1. Claims 10 depends from claim 9. The combination of PAE36 reference and Emmes is distinguished above with respect to the base claims 1 and 9 and similar limitations found in independent claim 13.

With respect to the additional limitations of claims 2, 3, 10, and 13, page 7 of the Official Action says: "PAE36 combined with Emmes fails to teach the common chunk of physical memory includes memory allocated to at least one processor stack, buffer cache, BIOS, and device drivers. Atherton col. 7, lines 2-20 is cited for showing a processor stack. Atherton col. 7, lines 7-9 say: "Generally, program modules include routines, programs, components (such as stacks or caches), data structures, etc., that perform particular tasks or implement particular abstract data types." Atherton col. 13 lines 1-12 are cited for showing a buffer cache. Atherton col. 13, lines 1-12 disclose that a block data structure is typically maintained in a virtual cache so that a routing engine module can access the information quickly without having to load the block of

memory again from map data. Atherton column 7, lines 26-31 are cited for disclosing BIOS. Atherton column 7, lines 26-31 say that BIOS is a basic input/output system stored in ROM. "The BIOS 26 essentially contains the basic routines that help to transfer information between elements within the personal computer 20 during certain computer operations, such as during start-up." Atherton col. 7, lines 55-59 are cited for showing device drivers. Atherton col. 7, lines 55-59 say: "The operating system 35, in conjunction with the BIOS 26 and associated device drivers, provides the basic interface between the computer's hardware and software resources, the user, and program modules, such as the map program module 36." Thus, Atherton disclose that a personal computer has processor stack, buffer cache, BIOS, and device drivers, but Atherton does not disclose that these elements should be included in the common chunk of physical memory. Nor is it seen where Atherton discloses that the common physical memory is at the bottom of the physical address space as recited in applicant's claim 10, or where Atherton discloses that the chunk of physical memory allocated to BIOS and device drivers is mapped to a top region of each of the plurality of virtual memory spaces as recited in applicant's claims 11 and 13.

On page 8 of the Official Action, claims 8, 12, 14, and 19-20 were rejected under 35 U.S.C. 103(a) as being unpatentable over the PAE36 reference and Emmes and Atherton et al. further in view of Perrin et al. (U.S. Patent 6,618,792). Applicant respectfully traverses. Claims 8, 12, 14, and 19-20 are dependent claims, and their respective base claims 1, 9, 13, and 15 are distinguished from the combination of PAE36, Emmes and Atherton as set out above.

Page 9 of the Official Action recognizes that neither PAE36, Emmes, nor Atherton teaches a DNLC. The Official Action cites Perrin col. 1, lines 57-64, which say:

> To minimize read operations on the persistent storage device, some operating systems have employed a directory name look-up cache (DNLC). As a central (global) resource, the directory name look-up cache provides information which can be used to locate most recently used files without having to perform read operations on the persistent storage device (e.g., disk). FIG. 1B depicts a computing environment 100 including a directory name look-up cache 102 suitable for storing filenames and references which provide access to files identified by the filenames. In order to provide access to a file, the operating system 104 first checks the directory name look-up cache 102 to determine whether the desired filename can be found. If the file name is not found in the directory name look-up cache 102, the operating system can initiate a search of a disk 106 to locate the file. Once the desired file is found, information about how to access the file can be cached into the directory name look-up cache 102 for future use.

Page 9 of the Official Action also cites Atherton column 13, lines 31-44, which say:

> In summary, the routing engine module 420 can complete its determination of the route by accessing map elements via information within the block data structure. Those skilled in the art will realize that once the block data structure is created for a particular region, the routing engine module 420 may not have to create another block data structure soon because the map elements needed may be from within a single region. Thus, it is advantageous to maintain several block data structures within the virtual memory cache 465 to avoid unnecessary reloading of blocks

> from the map data 430. The block data structure and the virtual memory cache 465 are both described in more detail below with regard to FIGS. 5 and 6, respectively.

With reference to claim 8, it is not seen where the combination of PAE36, Emmes, Atherton, and Perrin discloses that in a digital computer [programmed for moving data between network clients and data storage, there should be a first virtual address space containing an inode cache, a second virtual address space containing a dynamic name lookup cache for finding an inode having a given path name, and a third virtual address space containing a block map for snapshot copies. In other words, the combination of references does not suggest the particular assignment of virtual memory spaces to the particular well-defined or well-contained software modules executed by a processor of such a data mover computer.

With respect to claims 12 and 14, it is not seen where the combination of PAE36, Emmes, Atherton, and Perrin discloses that in a digital computer [programmed for moving data between network clients and data storage, there should be a first virtual address space directly mapped to a bottom region of the physical memory address space (see base claims 9 and 13), a second virtual address space containing a dynamic name lookup cache, and a third virtual address space containing a block map accessed by snapshot copy software. In other words, the combination of references does not suggest the particular assignment of virtual memory spaces to the particular well-defined or well-contained software modules executed by a processor of such a data mover computer.

With respect to claim 20, it is not seen where the combination of PAE36, Emmes, Atherton, and Perrin discloses that in a digital computer programmed for moving data between network clients and data storage, there should be a second virtual address space containing a dynamic name lookup cache (incorporated by reference from claim 19), and a third virtual address space containing a block map accessed by snapshot copy software. In other words, the combination of references does not suggest the particular assignment of virtual memory spaces to the particular well-defined or well-contained software modules executed by a processor of such a data mover computer.

In view of the above, reconsideration is respectfully requested, and early allowance is earnestly solicited.

Respectfully submitted,

Richard C. Auchterlonie

Richard C. Auchterlonie
Reg. No. 30,607

NOVAK DRUCE & QUIGG, LLP
1000 Louisiana, 53rd Floor
Houston, TX 77002
713-571-3460